CCP6114 Programming Fundamentals

Trimester 2430

ASSIGNMENT DOCUMENTATION

Light Mariadb Interpreter

Lecture Name: Mr. Goh Chien Le

Group: TC1L_TT2L_G01

NO	STUDENT ID	STUDENT NAME
1.	242UC244KZ	CHAN JIA HUI

Table of Contents

Pseudocodes

START

DECLARE functions:
- parseCSV(row)
- join(elements, delimiter)
- getColumnIndex(tableName, columnName)
- createFile(filename)
- createTable(line)
- insertIntoTable(line)
- displayTables()
- processLine(line, filename)
- displayDatabasePath(filename)
- writeToOutputFile(message)
- processDeleteData(deleteCommand)
- processUpdateData(updateCommand)

DECLARE data structures:
- createdTables (set) // stores names of created tables
- tableData (map) // stores data for each table
- tableSchemas (map) // stores schema for each table
- outputFile (ofstream) // file stream for writing output

FUNCTION parseCSV(row):
- SPLIT row by commas
- REMOVE extra spaces and quotes from each value
- REMOVE parentheses and special characters
- RETURN cleaned list of values

FUNCTION join(elements, delimiter):
- CONCATENATE elements with delimiter in between
- RETURN joined string

FUNCTION getColumnIndex(tableName, columnName):
- LOOKUP table schema in tableSchemas using tableName
- FIND columnName in schema
- IF not found, PRINT error message and RETURN invalid index
- ELSE, RETURN index of the column

FUNCTION writeToOutputFile(message):
- IF message is not empty:

- PRINT message to screen
- WRITE message to output file if open

FUNCTION createFile(filename):
- OPEN output file for writing
- IF file is successfully opened, PRINT message and WRITE to file
- ELSE, PRINT error message

FUNCTION createTable(line):
- EXTRACT table name from line
- IF table is not created yet, ADD to createdTables
- ELSE, PRINT table already exists message

FUNCTION insertIntoTable(line):
- EXTRACT table name from line
- IF table doesn't exist, PRINT error message
- EXTRACT and CLEAN values from line
- REMOVE parentheses, quotes, and semicolon from values
- STORE cleaned values in tableData for the respective table

FUNCTION displayTables():
- IF no tables created, PRINT no tables message
- ELSE, PRINT names of all created tables

FUNCTION displayDatabasePath(filename):
- PRINT the path of the database file

FUNCTION processDeleteData(deleteCommand):
- EXTRACT table name and WHERE condition from DELETE command
- CHECK if table exists
- DELETE rows that match WHERE condition in tableData

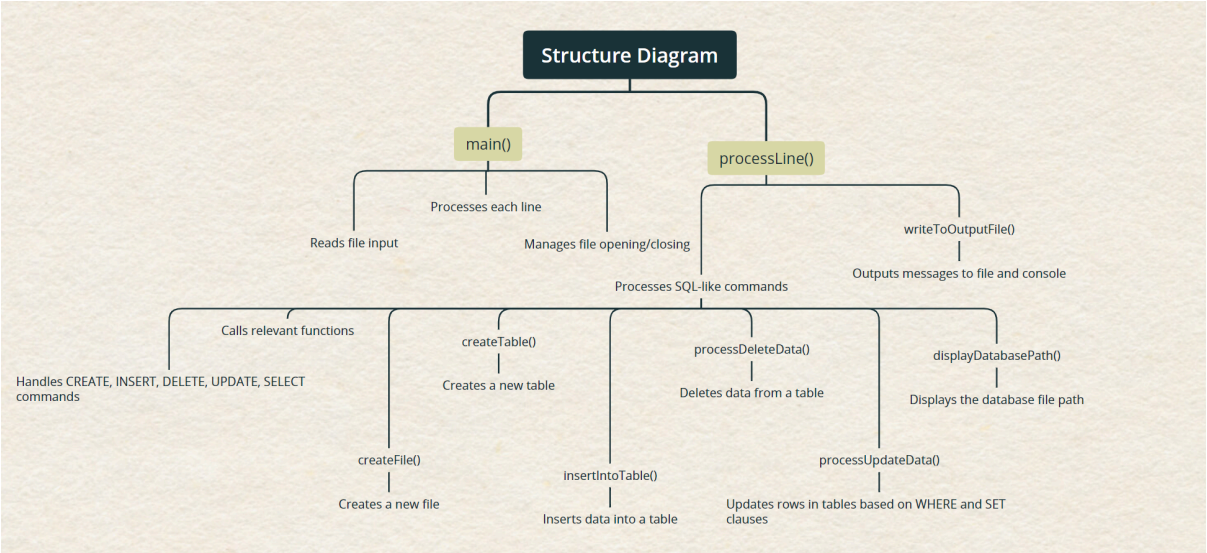
FUNCTION processUpdateData(updateCommand):
- EXTRACT table name, SET clause, and WHERE condition from UPDATE command
- IF any error in the command, PRINT error message
- GET column indices for update and WHERE column
- UPDATE rows matching WHERE condition

FUNCTION processLine(line, filename):
- IF line is empty or whitespace-only, SKIP it
- IF line is a command (CREATE, INSERT, etc.), MODIFY line to add '>' before command and write to file
- CALL appropriate functions based on the command type (CREATE, INSERT, SELECT, UPDATE, DELETE)

MAIN:
- SET basePath to directory path
- GET filename from user input
- OPEN file for reading
- IF file fails to open, PRINT error message
- ELSE, READ each line and process it using processLine
- CLOSE input file
- CLOSE output file after processing

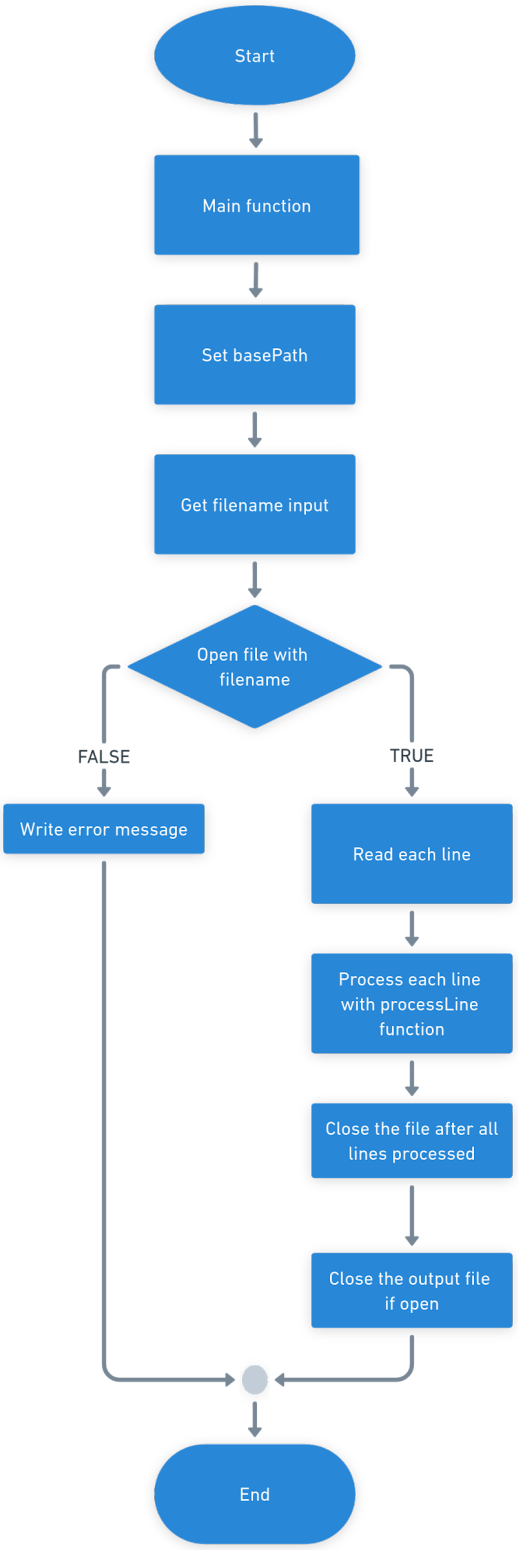
END

Structure Diagrams

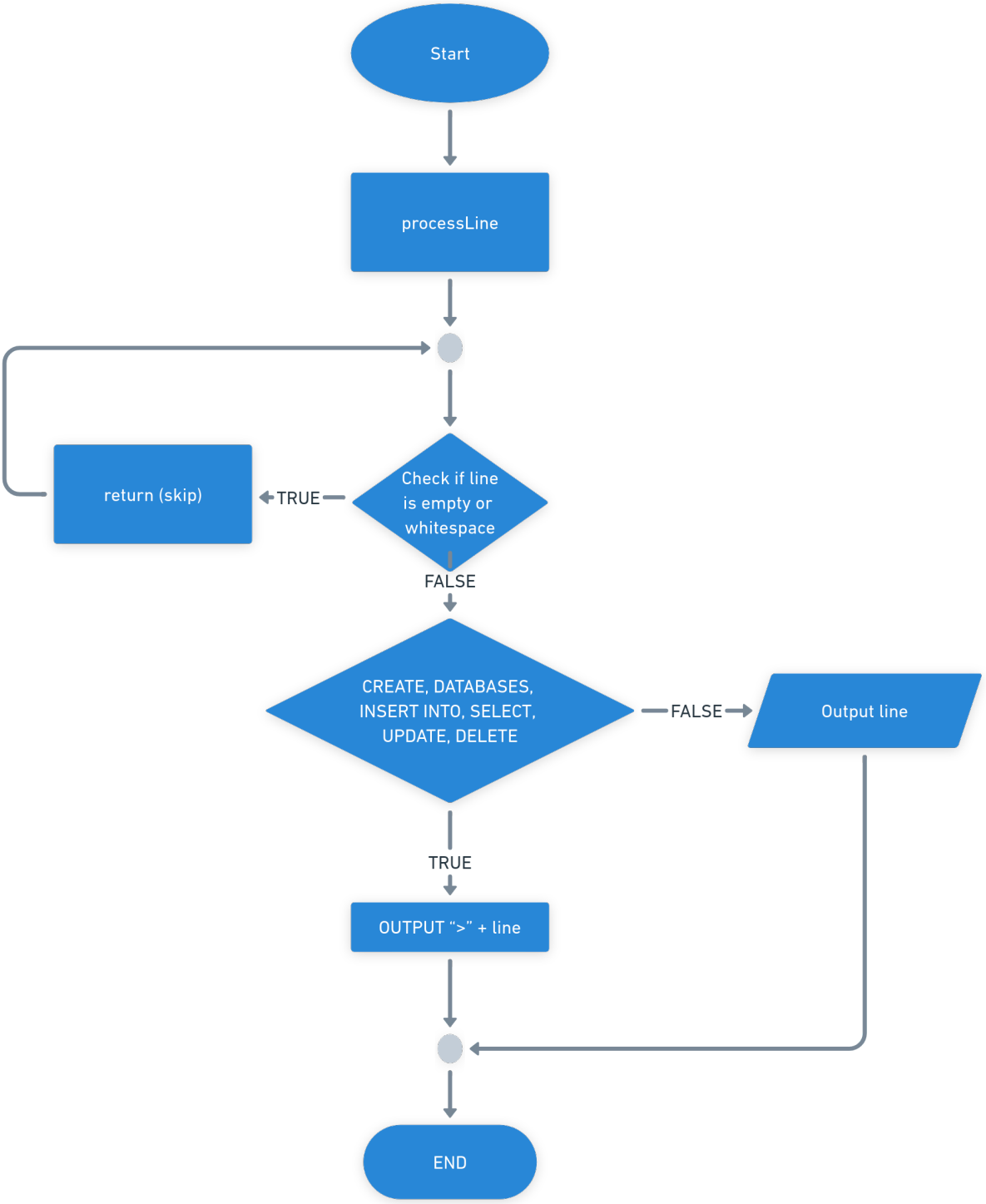


Flowchart

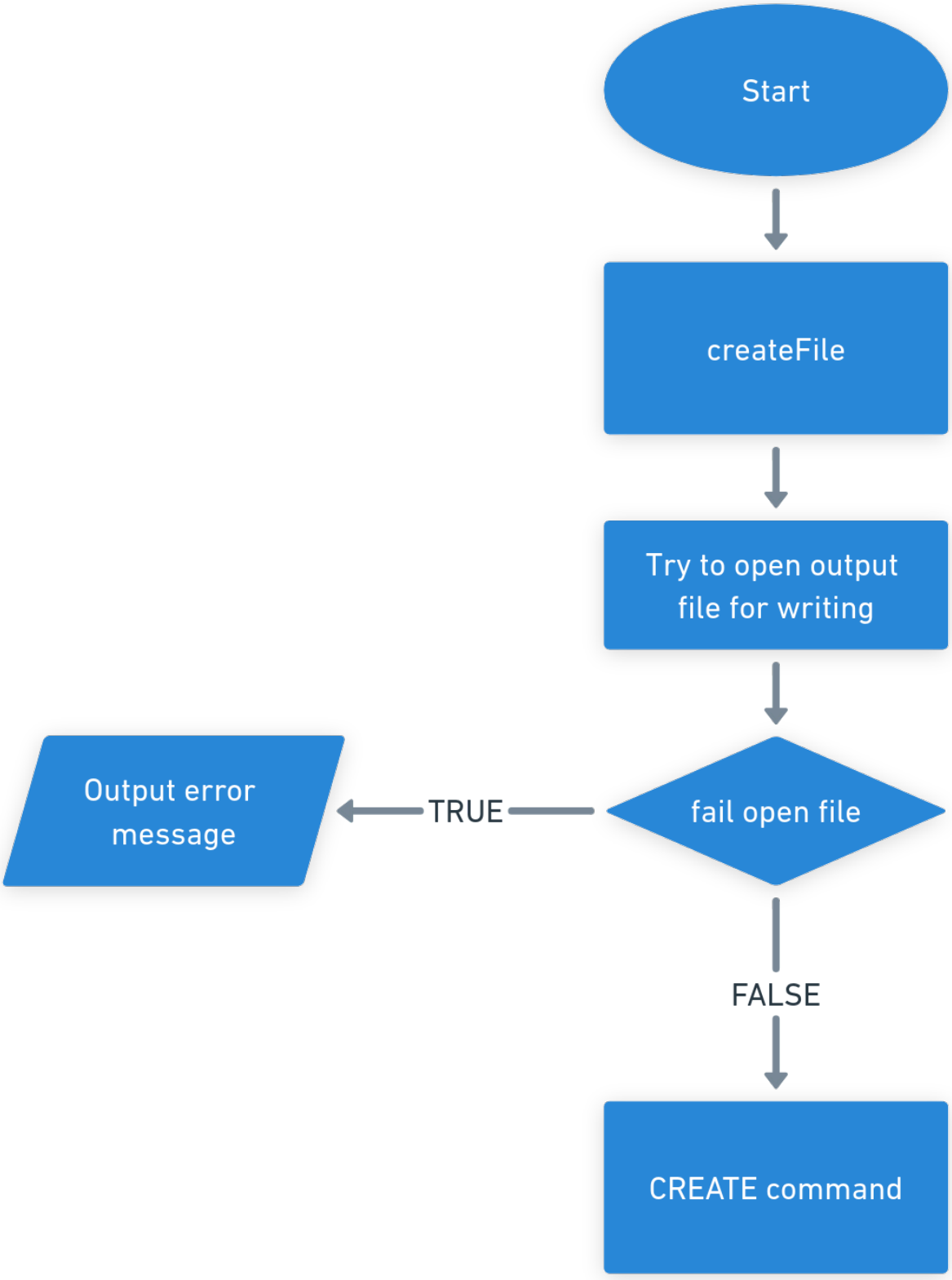
main function



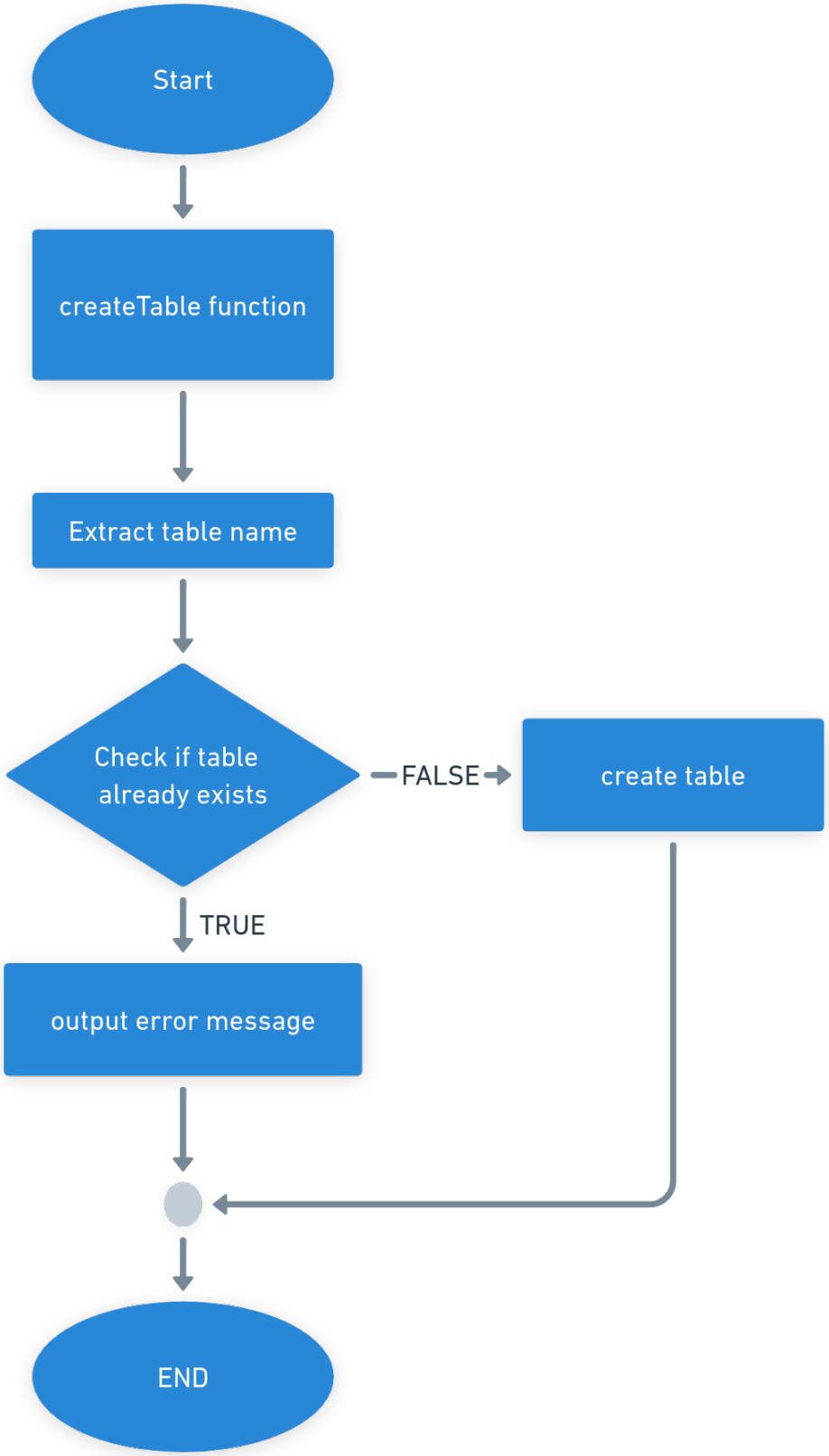
processLine function



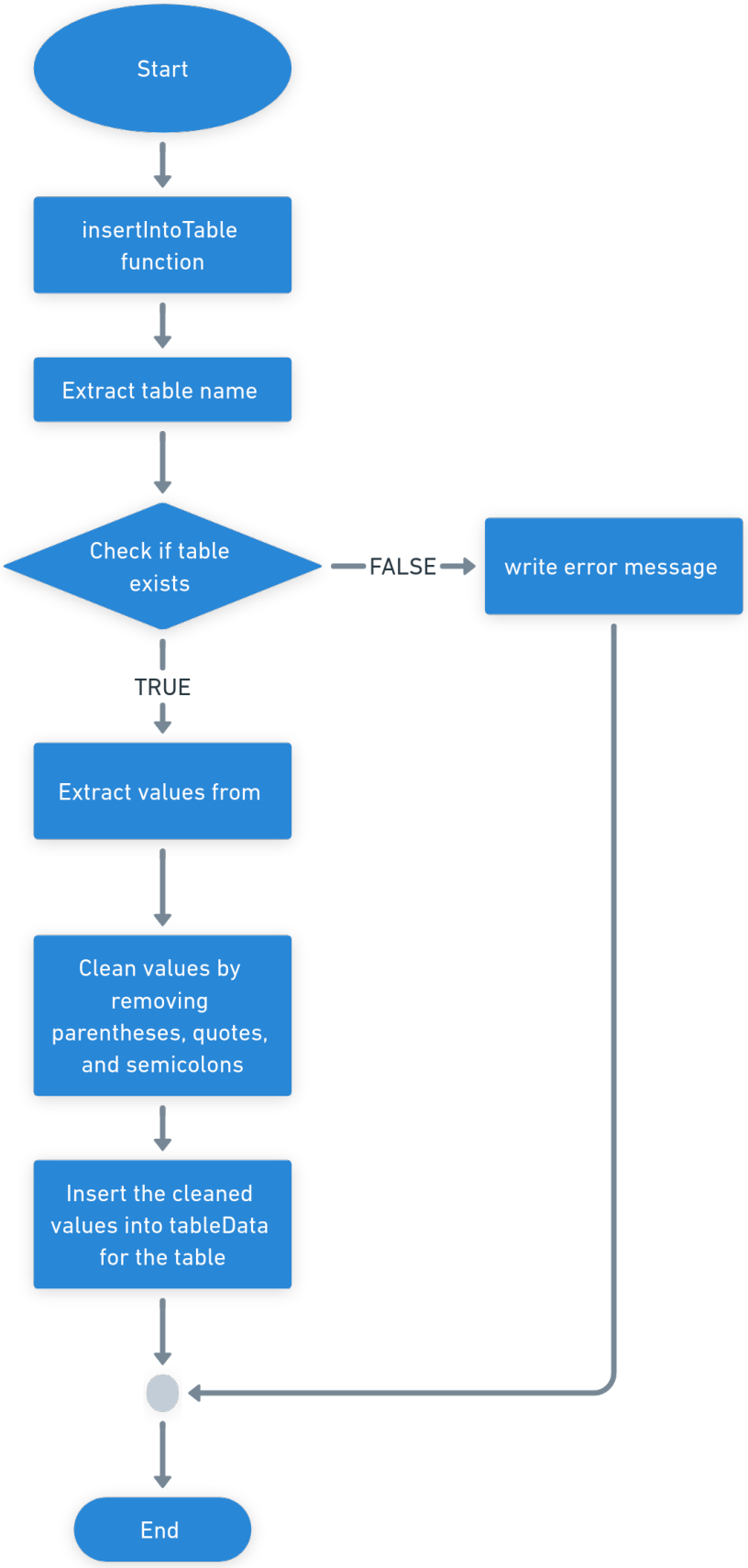
createFile function



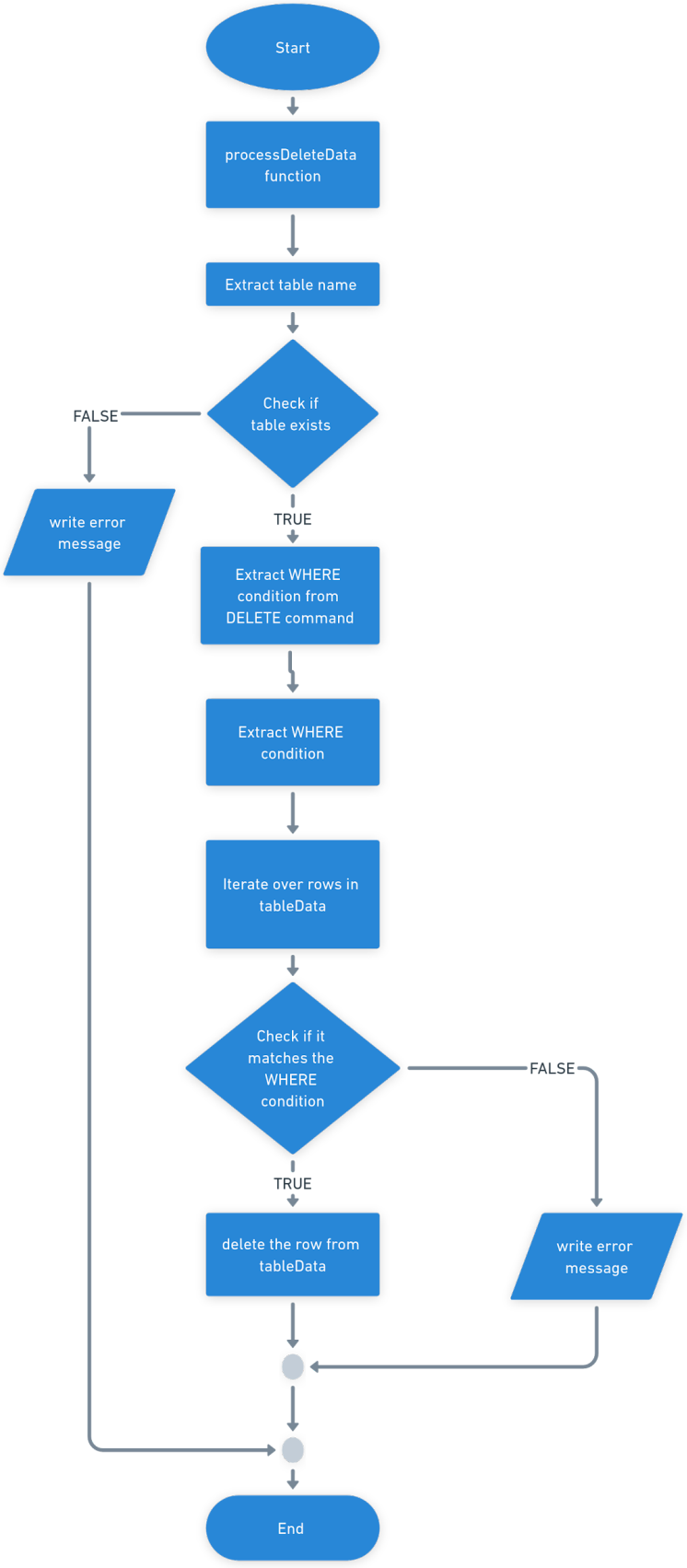
createTable function



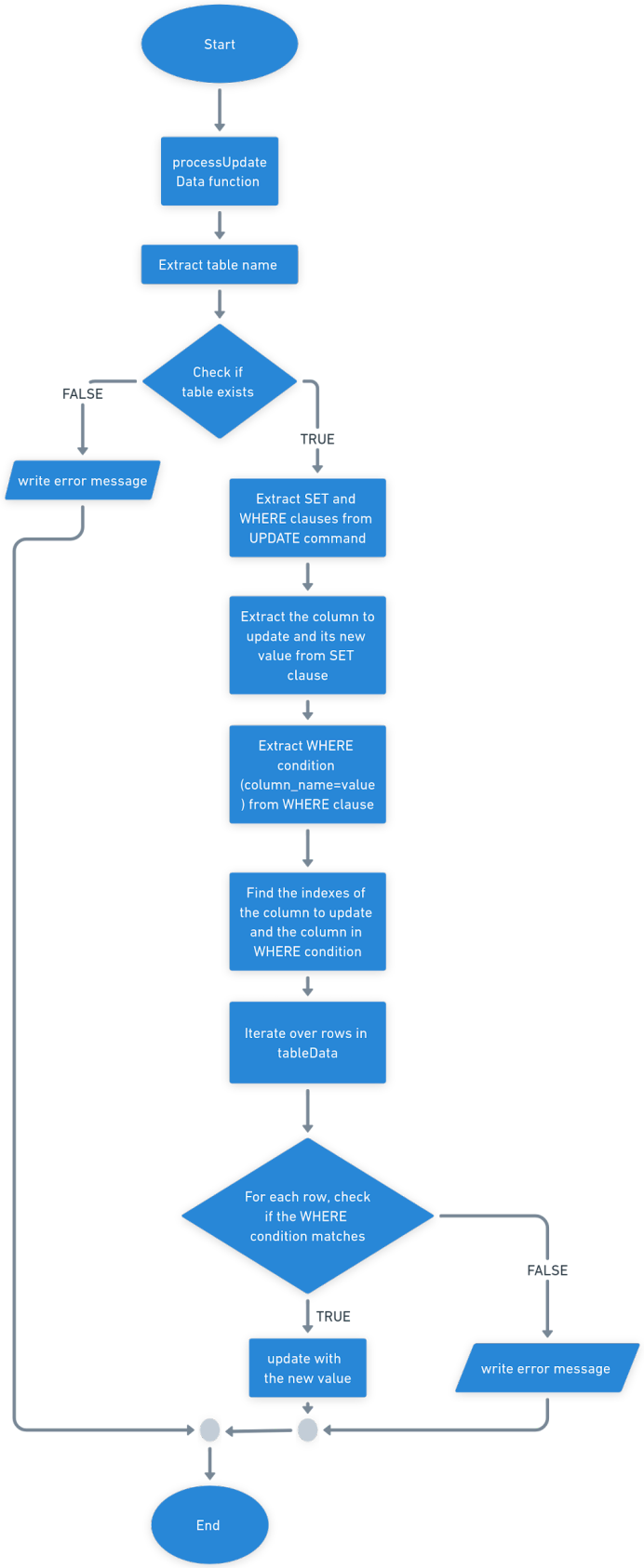
insertIntoTable function



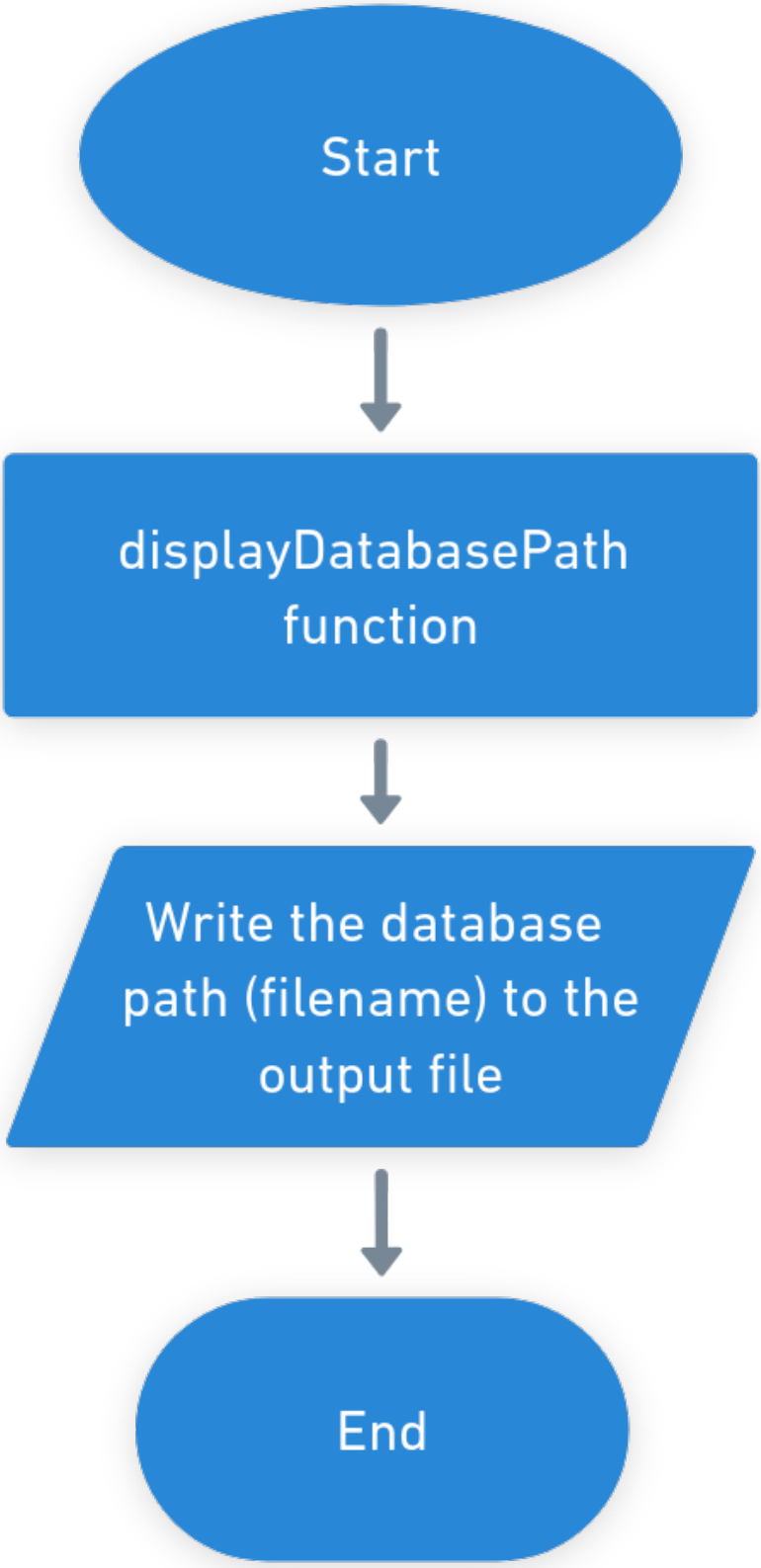
processDeleteData function



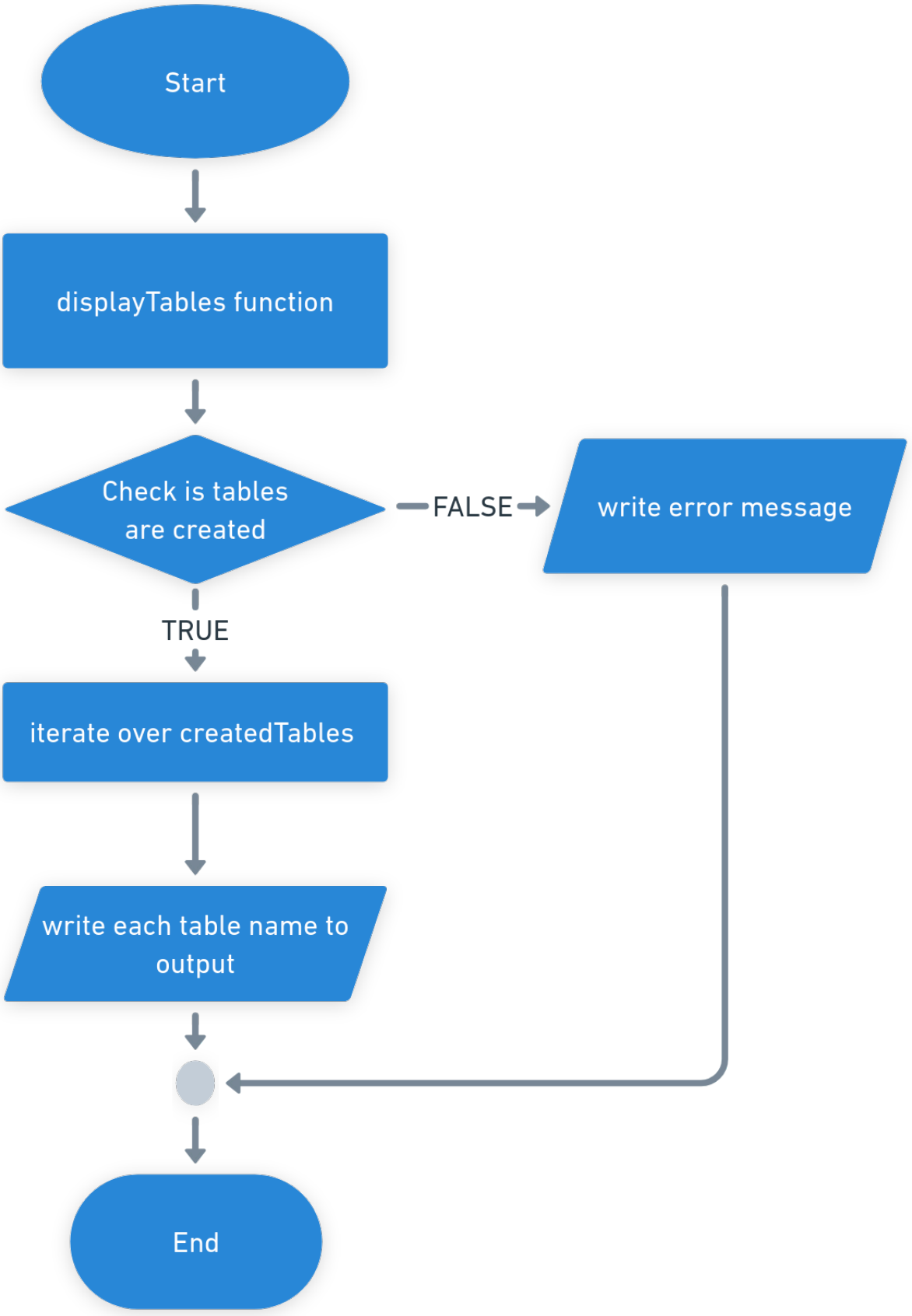
processUpdateData function



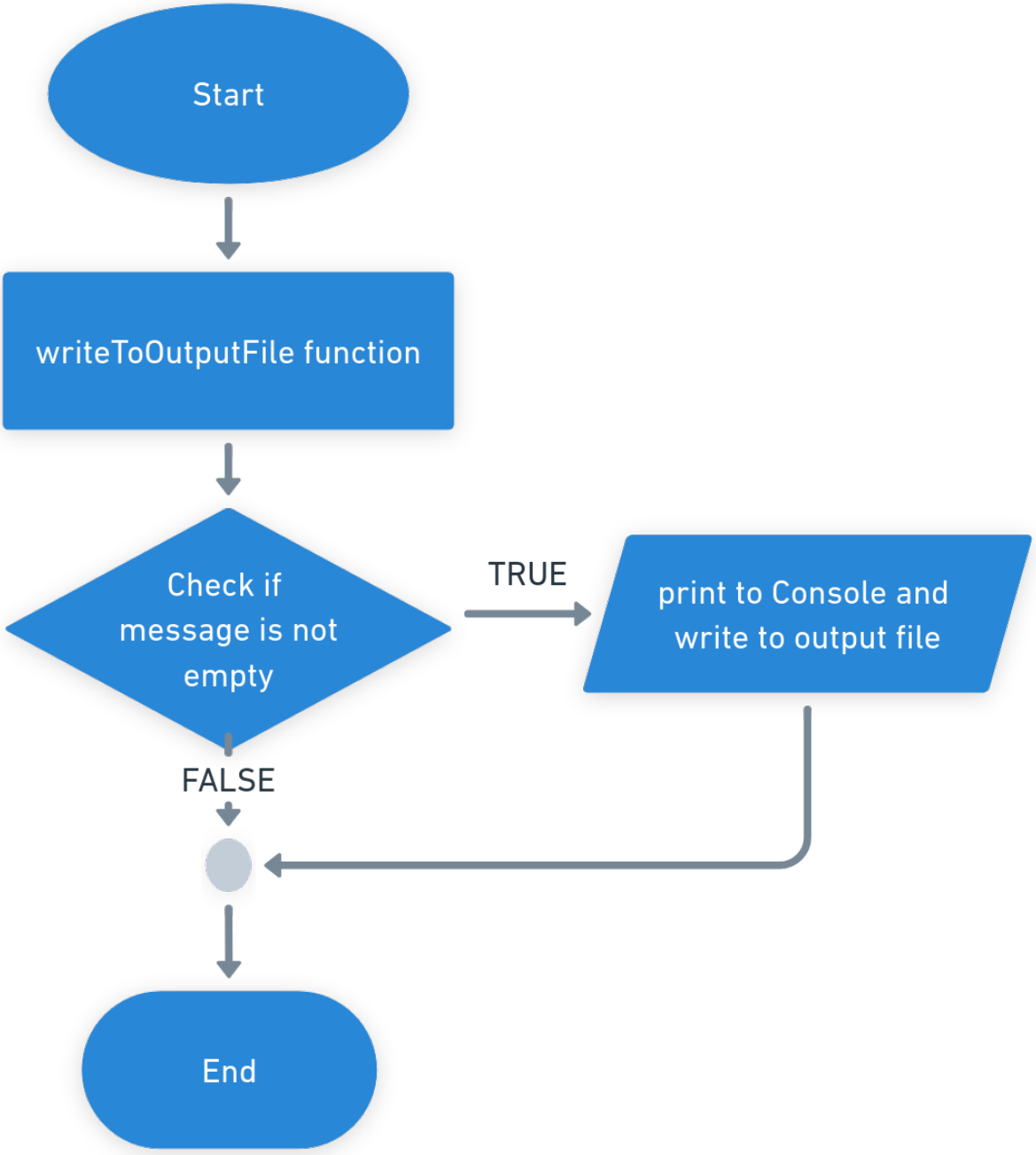
displayDatabasePath function



displayTables function



writeToOutputFile function



SampleInput1

fileInput1.mdb

```
CREATE fileOutput1.txt;
DATABASES;

CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);

TABLES;

INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (1,'name1','city1','state1','country1','phone1','email1');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (2,'name2','city2','state2','country2','phone2','email2');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (3,'name3','city3','state3','country3','phone3','email3');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (4,'name4','city4','state4','country4','phone4','email4');


SELECT * FROM customer;
```

screen output1

```
fileInput1.mdb
>CREATE fileOutput1.txt;
>CREATE fileOutput1.txt;
>DATABASES;
C:\Users\User\Projects\Light_Mariadb_Interpreter\data\fileInput1.mdb
>CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);
>TABLES;
customer
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (1,'name1','city1','stat
e1','country1','phone1','email1');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (2,'name2','city2','stat
e2','country2','phone2','email2');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (3,'name3','city3','stat
e3','country3','phone3','email3');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (4,'name4','city4','stat
e4','country4','phone4','email4');
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email3
4,name4,city4,state4,country4,phone4,email4
```

fileOutput1.txt

```
>CREATE fileOutput1.txt;
>DATABASES;
C:\Users\User\Projects\Light_Mariadb_Interpreter\data\fileInput1.mdb
>CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);
>TABLES;
customer
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (1,'name1','city1','state1','country1','phone1','email1');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (2,'name2','city2','state2','country2','phone2','email2');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (3,'name3','city3','state3','country3','phone3','email3');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (4,'name4','city4','state4','country4','phone4','email4');
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email3
4,name4,city4,state4,country4,phone4,email4
```

SampleInput2

fileInput2.mdb

```
CREATE fileOutput2.txt;
DATABASES;

CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);

INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (1,'name1','city1','state1','country1','phone1','email1');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (2,'name2','city2','state2','country2','phone2','email2');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (3,'name3','city3','state3','country3','phone3','email3');
INSERT INTO
customer(customer_id,customer_name,customer_city,customer_state,customer_country,custo
mer_phone,customer_email) VALUES (4,'name4','city4','state4','country4','phone4','email4');

SELECT * FROM customer;

TABLES;

UPDATE customer SET customer_email='email333' WHERE customer_id=3;
SELECT * FROM customer;

DELETE FROM customer WHERE customer_id=4;
SELECT * FROM customer;

SELECT COUNT(*) FROM customer;
```

screen output2

```
fileInput2.mdb
>CREATE fileOutput2.txt;
>CREATE fileOutput2.txt;
>DATABASES;
C:\Users\User\Projects\Light_Mariadb_Interpreter\data\fileInput2.mdb
>CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (1,'name1','city1','state1','country1','phone1
','email1');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (2,'name2','city2','state2','country2','phone2
','email2');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (3,'name3','city3','state3','country3','phone3
','email3');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (4,'name4','city4','state4','country4','phone4
','email4');
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email3
4,name4,city4,state4,country4,phone4,email4
>TABLES;
customer
>UPDATE customer SET customer_email='email333' WHERE customer_id=3;
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email333
4,name4,city4,state4,country4,phone4,email4
>DELETE FROM customer WHERE customer_id=4;
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email333
>SELECT COUNT(*) FROM customer;
3
```

fileOutput2.txt

```
>CREATE fileOutput2.txt;
>DATABASES;
C:\Users\User\Projects\Light_Mariadb_Interpreter\data\fileInput2.mdb
>CREATE TABLE customer(
customer_id INT,
customer_name TEXT,
customer_city TEXT,
customer_state TEXT,
customer_country TEXT,
customer_phone TEXT,
customer_email TEXT
);
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (1,'name1','city1','state1','country1','phone1','email1');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (2,'name2','city2','state2','country2','phone2','email2');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (3,'name3','city3','state3','country3','phone3','email3');
>INSERT INTO customer(customer_id,customer_name,customer_city,customer_state,customer_country,customer_phone,customer_email) VALUES (4,'name4','city4','state4','country4','phone4','email4');
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email3
4,name4,city4,state4,country4,phone4,email4
>TABLES;
customer
>UPDATE customer SET customer_email='email333' WHERE customer_id=3;
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email333
4,name4,city4,state4,country4,phone4,email4
>DELETE FROM customer WHERE customer_id=4;
>SELECT * FROM customer;
customer_id, name, email, phone, address
1,name1,city1,state1,country1,phone1,email1
2,name2,city2,state2,country2,phone2,email2
3,name3,city3,state3,country3,phone3,email333
>SELECT COUNT(*) FROM customer;
3
```

SampleInput3

fileInput3.mdb

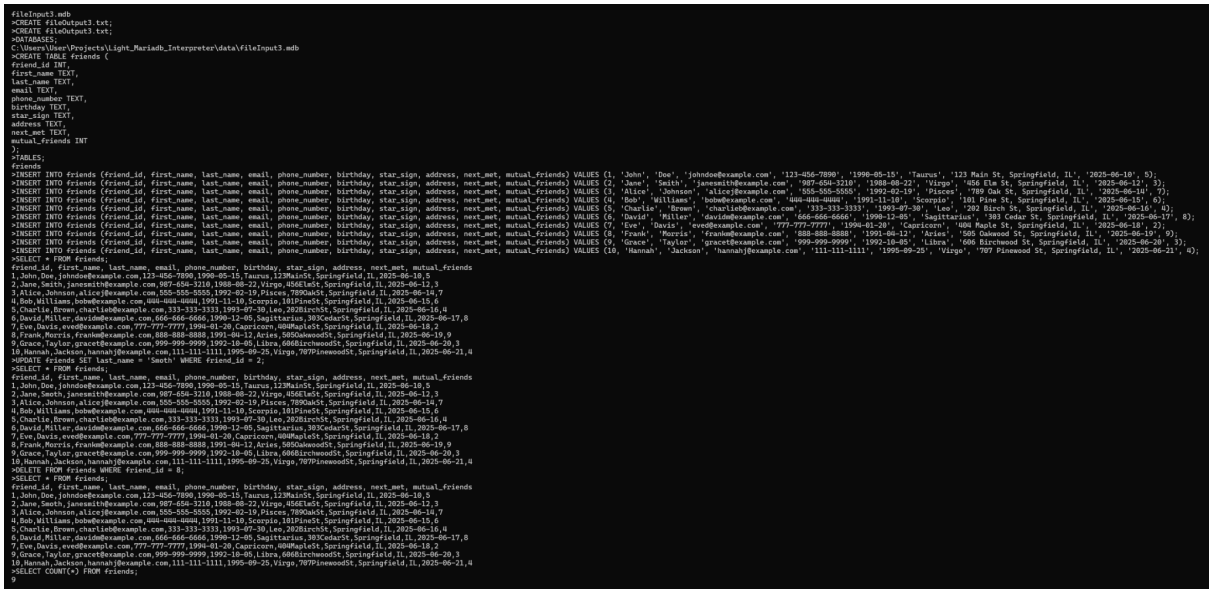
CREATE fileOutput3.txt;
DATABASES;

CREATE TABLE friends (
friend_id INT,
first_name TEXT,
last_name TEXT,
email TEXT,
phone_number TEXT,
birthday TEXT,
star_sign TEXT,
address TEXT,
next_met TEXT,
mutual_friends INT
);
TABLES;

INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (1, 'John', 'Doe', 'johndoe@example.com', '123-456-7890', '1990-05-15', 'Taurus', '123 Main St, Springfield, IL', '2025-06-10', 5);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (2, 'Jane', 'Smith', 'janesmith@example.com', '987-654-3210', '1988-08-22', 'Virgo', '456 Elm St, Springfield, IL', '2025-06-12', 3);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (3, 'Alice', 'Johnson', 'alicej@example.com', '555-555-5555', '1992-02-19', 'Pisces', '789 Oak St, Springfield, IL', '2025-06-14', 7);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (4, 'Bob', 'Williams', 'bobw@example.com', '444-444-4444', '1991-11-10', 'Scorpio', '101 Pine St, Springfield, IL', '2025-06-15', 6);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (5, 'Charlie', 'Brown', 'charlieb@example.com', '333-333-3333', '1993-07-30', 'Leo', '202 Birch St, Springfield, IL', '2025-06-16', 4);

```
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (6, 'David', 'Miller', 'davidm@example.com', '666-666-6666', '1990-12-05', 'Sagittarius', '303 Cedar St, Springfield, IL', '2025-06-17', 8);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (7, 'Eve', 'Davis', 'eved@example.com', '777-777-7777', '1994-01-20', 'Capricorn', '404 Maple St, Springfield, IL', '2025-06-18', 2);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (8, 'Frank', 'Morris', 'frankm@example.com', '888-888-8888', '1991-04-12', 'Aries', '505 Oakwood St, Springfield, IL', '2025-06-19', 9);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (9, 'Grace', 'Taylor', 'gracet@example.com', '999-999-9999', '1992-10-05', 'Libra', '606 Birchwood St, Springfield, IL', '2025-06-20', 3);
INSERT INTO friends (friend_id, first_name, last_name, email, phone_number, birthday, star_sign, address, next_met, mutual_friends) VALUES (10, 'Hannah', 'Jackson', 'hannahj@example.com', '111-111-1111', '1995-09-25', 'Virgo', '707 Pinewood St, Springfield, IL', '2025-06-21', 4);

SELECT * FROM friends;

UPDATE friends SET last_name = 'Smoth' WHERE friend_id = 2;
SELECT * FROM friends;

DELETE FROM friends WHERE friend_id = 8;
SELECT * FROM friends;

SELECT COUNT(*) FROM friends;
```

screen output3



fileOutput3.txt

Explanation

1. Header Files and Namespaces
    a. <iostream>: For input-output operations like reading from cin and writing to cout.
    b. <sstream>: For working with string streams (used in parsing CSV rows).
    c. <fstream>: For reading and writing files.
    d. <string>: For handling strings.
    e. <unordered_set>, <unordered_map>: For storing tables and data.
    f. <vector>: For storing collections of strings.
    g. <algorithm>: For algorithms like find and remove.
    h. using namespace std; allows us to use standard C++ classes and functions without needing to prefix them with std::.
2. Function Prototypes
    a. The function prototypes define the functions we will use later to handle the operations like parsing CSV data, creating tables, inserting data, updating data, and others.
3. parseCSV Function
    a. This function is responsible for parsing a CSV row and returning the individual values as a vector of strings.
    b. It removes spaces, quotes, and parentheses that might surround the data. This is important for cleanly processing the values in SQL commands.
4. join Function
    a. Joins the elements of a vector into a single string, separated by the specified delimiter.
5. Global Variables
    a. createdTables: Holds the names of tables that have been created.
    b. tableData: Stores the data of tables, with the table name as the key.
    c. tableSchemas: Defines the schema (columns) for predefined tables like friends and customer.
6. getColumnIndex Function
    a. Finds the index of a column in a table's schema. If the column does not exist, it returns string::npos.
7. writeToOutputFile Function
    a. Writes messages to both the console and an output file. If the message is non-empty, it will be displayed on screen and saved to the output file.
8. createFile Function
    a. Creates an output file for storing processed data or SQL statements. If the file cannot be created, it logs an error.
9. createTable Function
    a. Extracts the table name from a CREATE TABLE command and adds it to createdTables if not already created. If the table exists, it logs an error.

10. insertIntoTable Function
    a. Extracts the table name and values from an INSERT INTO command and stores the row in tableData.
11. displayTables Function
    a. Displays all the created table names.
12. displayDatabasePath Function
    a. Displays the path of the database file.
13. processDeleteData Function
    a. Handles the DELETE FROM SQL command by extracting the table name and WHERE condition to delete matching rows.
14. processUpdateData Function
    a. Handles the UPDATE SQL command by modifying rows based on the specified SET and WHERE conditions.
15. processLine Function
    a. This is the main function to process each line of input from the file.
    b. It checks the type of command (CREATE, INSERT INTO, SELECT, UPDATE, DELETE, etc.) and delegates the task to the appropriate helper functions.
16. main Function
    a. It opens the input file and reads it line by line.
    b. Each line is passed to processLine() for execution.
    c. After processing, it closes the output file.

Program Output

1. Commands with >: When executing a SQL-like command (like CREATE, INSERT, SELECT), it adds a > symbol to the command and writes it to the output file.
2. Table Creation: When a table is created, it logs a message confirming the table creation.
3. Database Path: When the DATABASES command is executed, the path of the database file is written to the output file.
4. Inserting Data: When INSERT INTO is processed, the inserted values are added to the corresponding table's data.
5. Table Data Display: When a SELECT * FROM table command is executed, the program displays the table data in a CSV-like format (without the parentheses and quotes).
6. Row Count in Table: If the SELECT COUNT(*) FROM table command is executed, the program will output the number of rows in the specified table.
7. Delete and Update Operations: The DELETE and UPDATE commands remove or modify rows in the corresponding table based on the specified conditions.

User Documentation

1. Input Commands
    a. Create Command: This command creates a new table in the database.
    b. Insert Command: This command inserts data into an existing table in the database.
    c. Select Command: This command retrieves data from a table. It can fetch all columns or count the number of rows in a table.
    d. Update Command: This command updates existing records in a table based on a condition.
    e. Delete Command: This command deletes records from a table based on a condition.
    f. Database Command: This command displays the path of the database file.
    g. Tables Command: This command shows all the created tables in the database.
2. Command Execution Flow
    a. Loading Commands: The interpreter reads SQL commands from an input file.
    b. Processing Commands: Each command is executed based on its type (e.g., CREATE, INSERT, SELECT).
    c. Logging Output: Executed commands and results are logged in the output.
    d. File Creation: An output file is created to record the commands and data.
3. Error Handling
    a. The system will report errors for missing or incorrect commands.
    b. Attempting to create a table that already exists will result in an error.
    c. If an invalid column is referenced in a command, an error message will be shown.